同济大学交通运输工程学院
COLLEGE OF TRANSPORTATION ENGINEERING
TONGJI UNIVERSITY

交通数据分析

第八讲 关联分析：频繁模式挖掘

沈煜 博士 副教授
嘉定校区交通运输工程学院311室
yshen@tongji.edu.cn
2022年04月15日

计划进度

周	日期	主讲	内容	模块
1	2022.02.25	沈煜	概述	
2	2022.03.04	沈煜	在线数据采集方法	爬虫
3	2022.03.11	沈煜	线性回归模型	回归分析
5	2022.03.18	沈煜	广义线性回归	
4	2022.03.25	沈煜	广义线性回归（作业1）	
6	2022.04.01	沈煜	空间数据描述性分析	
7	2022.04.08	沈煜	空间自回归方法（作业2）	
8	2022.04.15	沈煜	关联：Apriori	机器学习
9	2022.04.22	沈煜	决策树、支持向量机（作业3）	
10	2022.04.29	沈煜	浅层神经网络	
11	2022.05.06	沈煜	卷积神经网络（期末大作业）	
12	2022.05.13	沈煜	经典网络结构	
13	2022.05.20	沈煜	聚类：K-Means、DBSCAN	
14	2022.05.27	沈煜	贝叶斯方法、卡尔曼滤波	
15	2022.06.03	-	端午节放假	
16	2022.06.10	沈煜	期末汇报（1）	
17	2022.06.17	沈煜	期末汇报（2）	

频繁模式挖掘：基本概念和方法

- 基本概念
- 频繁项集挖掘方法
- 关联模式评估方法

同济大学交通运输工程学院
COLLEGE OF TRANSPORTATION ENGINEERING
TONGJI UNIVERSITY

基本概念

频繁模式（Frequent Pattern）

频繁模式分析的概念

- **频繁模式**（frequent pattern）
  - 频繁地出现在数据集中的模式（如项集、子序列或子结构）
- 由Agrawal等于1994年首次提出
- **目的：**解析数据的内在规律
  - 哪些商品经常会被同时购买？（例如：啤酒和尿布）
  - 顾客购买了电脑之后经常会再购买哪些商品?
  - 哪类DNA对某种新药物的反应敏感?
  - 我们能否对网页内容做自动分类?
- **应用场景**
  - 购买数据分析、交叉市场营销、商品目录设计、销售活动分析、上网浏览记录分析、DNA序列分析等。

频繁模式发掘的重要性

- **频繁模式：** 数据集内在的与重要的特性。
- 频繁模式挖掘是很多数据挖掘方法的重要基础
  - 关联性、相关性、因果分析
  - 序列、结构（如子图）模式分析
  - 时空数据、多媒体数据、时序数据、流数据中的模式分析
  - 分类: discriminative, frequent pattern analysis
  - 聚类：基于频繁模式的聚类
  - 数据仓库：iceberg cube and cube-gradient
  - 语义数据压缩: fascicles
  - 更广泛的应用

关联规则基本模型

- 设$I = \{i_1, \ldots, i_m\}$为所有项目的集合，D为事务数据库，事务T是一个项目子集（$T \subseteq I$）。每一个事务具有唯一的事务标识TID。
- 设A是一个由项目构成的集合，称为**项集**。事务T包含项集A，当且仅当 $A \subseteq T$ 。如果项集A中包含k个项目，则称其为***k*项集**。
- 项集A在事务数据库D中出现的次数占D中总事务的百分比叫做项集**支持度**。
- 如果项集的支持度超过用户给定的**最小支持度阈值**，就称该项集是**频繁项集**（或**大项集**）。

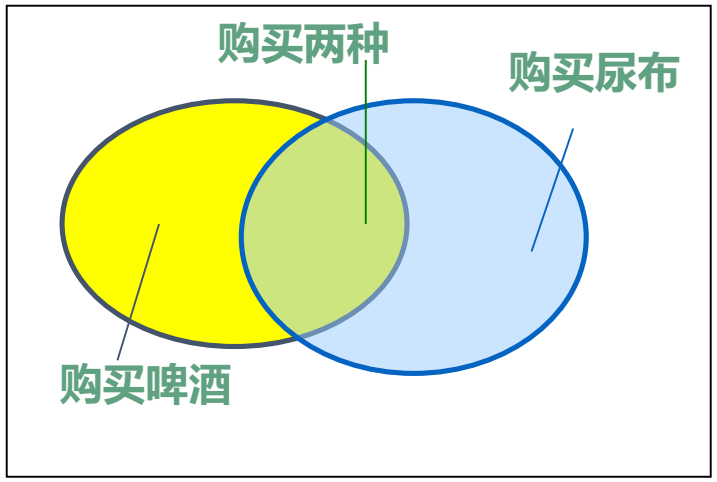
TID	购买的产品
10	啤酒, 花生, 尿布
20	啤酒, 咖啡, 尿布
30	啤酒, 尿布, 鸡蛋
40	花生, 鸡蛋, 牛奶
50	花生, 咖啡, 尿布, 鸡蛋, 牛奶

关联规则基本模型

- 关联规则是形如$X \Rightarrow Y$的逻辑蕴含式，其中$X \subset I$，$Y \subset I$，且$X \cap Y = \emptyset$。
- 如果事务数据库D中有$s\%$的事务包含$X \cup Y$，则称关联规则$X \Rightarrow Y$的**支持度为$s\%$**。
  - 实际上，支持度是一个概率值，是一个相对计数。
  - $support(X \Rightarrow Y) = P(X \cup Y)$
- 项集的**支持度计数（频率）support_count**
  - 包含项集的事务数
- 若项集X的**支持度**记为$support(X)$，规则的**置信度**为$support(X \cup Y)/support(X)$
  - 是一个条件概率$P(Y|X)$
  - $confidence(X \Rightarrow Y) = P(Y|X) = \frac{support\_count(X \cup Y)}{support\_count(X)}$

基本概念：关联规则

TID	购买的产品
10	啤酒, 花生, 尿布
20	啤酒, 咖啡, 尿布
30	啤酒, 尿布, 鸡蛋
40	花生, 鸡蛋, 牛奶
50	花生, 咖啡, 尿布, 鸡蛋, 牛奶



- 项集$X = \{x_1, \ldots, x_k\}$
- 找出满足最小支持度和置信度的所规则 $X \rightarrow Y$
  - **支持度**, s, 事务包含$X \cup Y$ 的**概率**
  - **置信度**, c, 事务含X也包含Y的**条件概率**
- 令$sup_{min} = 50\%$，$conf_{min} = 50\%$
- 频繁模式：
  - 啤酒：3，花生：3，尿布：4，鸡蛋：3，{啤酒，尿布}：3
- 关联规则:
  - 啤酒 → 尿布
  - (支持度60%, 置信度100%)
  - 尿布 → 啤酒
  - (支持度60%, 置信度75%)

挖掘关联规则：示例

TID	购买的商品
10	A, B, C
20	A, C
30	A, D
40	B, E, F

频繁模式	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

- 设最小支持度50%；最小置信度50%
- 规则$A \Rightarrow C$：
  - 支持度：$support\{\{A\} \cup \{C\}\} = 50\%$
  - 置信度：$support\{\{A\} \cup \{C\}\}/support\{\{A\}\} = 66.6\%$

同济大学交通运输工程学院
COLLEGE OF TRANSPORTATION ENGINEERING
TONGJI UNIVERSITY

频繁项集挖掘方法

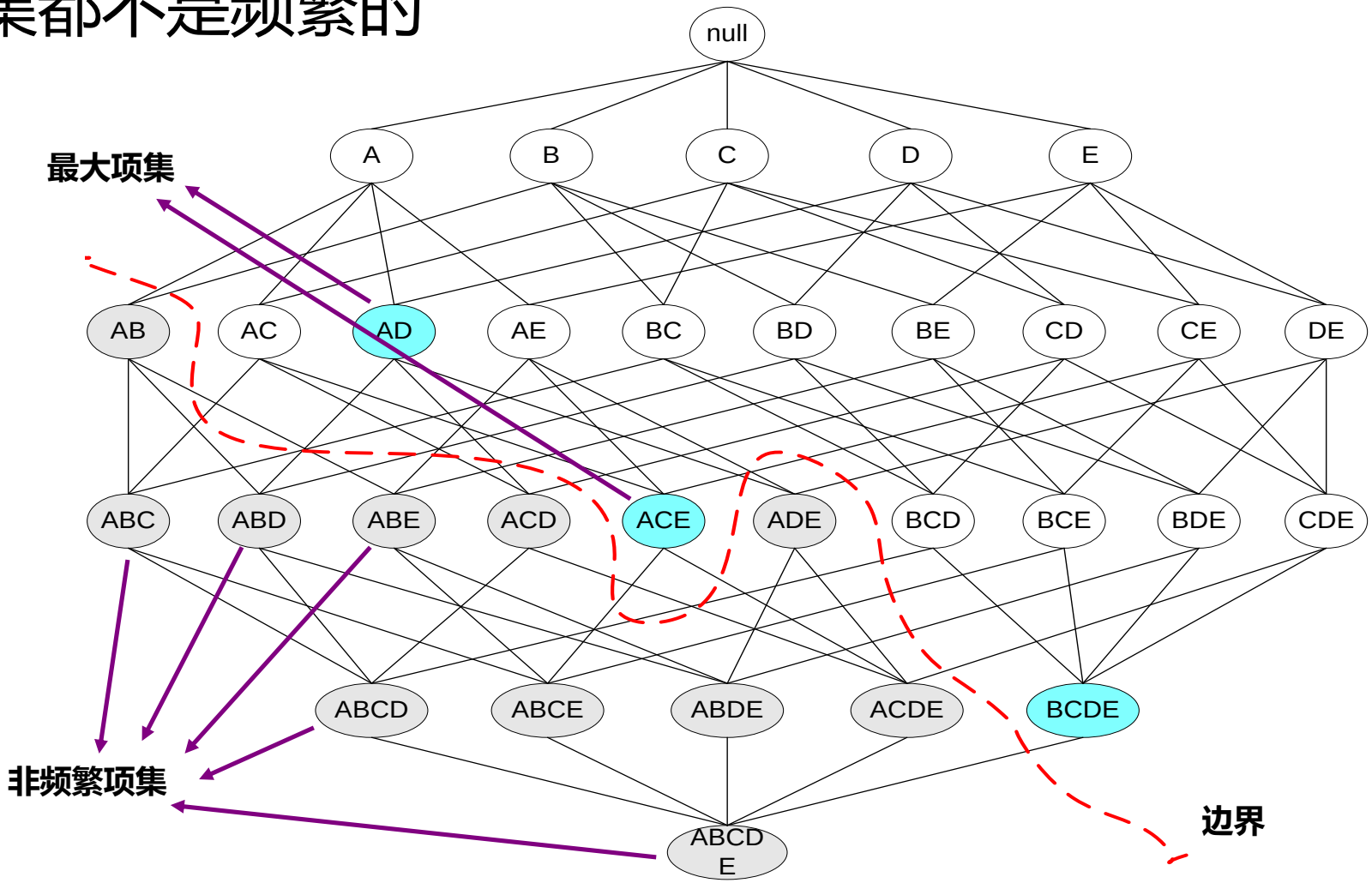
频繁项集

闭频繁项集和极大频繁项集

- 一个长模式包含子模式的数目：
  - 例如长度为100，$\{a_1, \ldots, a_{100}\}$，包含
  - $C_{100}^1 + C_{100}^2 + \cdots + C_{100}^{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$个子模式
- 解决方法：挖掘闭频繁项集和极大频繁项集
- 如果X是频繁的，且不存在X的真超项集Y使得Y与X在D中具有相同的支持度计数，那么频繁项集X在数据集D中是闭的。
  - 注：Y是X的真超项集（super-pattern）：X的每一项都包含在Y中（$X \subset Y$），但是Y中至少有一个项不在X中。
- 极大频繁项集：如果X是频繁的，且不存在超项集Y，使得$X \subset Y$并且Y在D中是频繁的。
- 两者有不同，极大频繁项集定义中对真超集要松一些。

极大频繁项集

- 如果一个项集是极大频繁的，那么它的所有直接超项集都不是频繁的



闭频繁项集

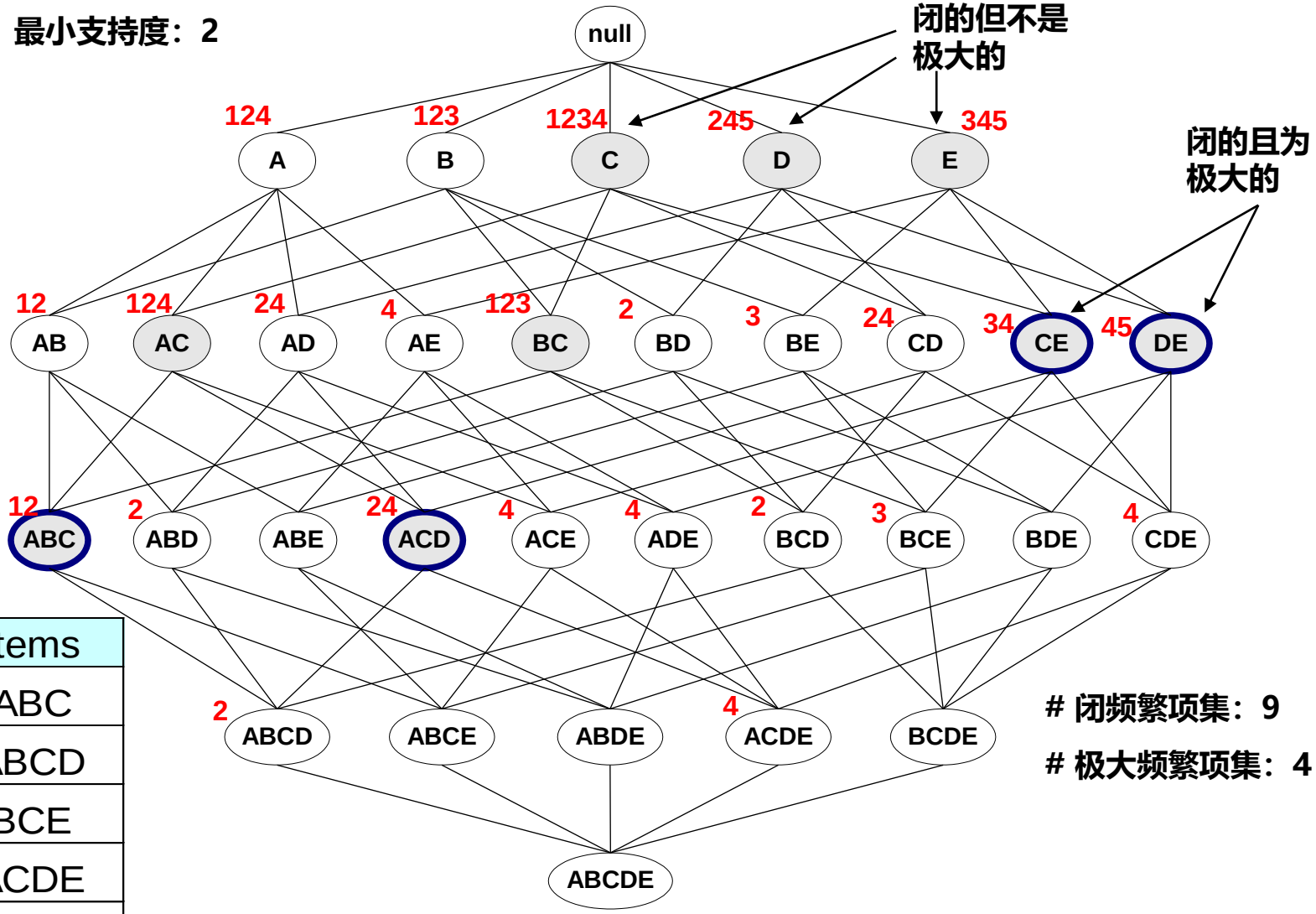
- 极大频繁项集的问题：
  - 其子项集的支持度是未知的，需要对数据库进行额外的扫描
- 如果一个项集是闭的，那么它的所有直接超项集都不具有和该项集同样的支持度

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

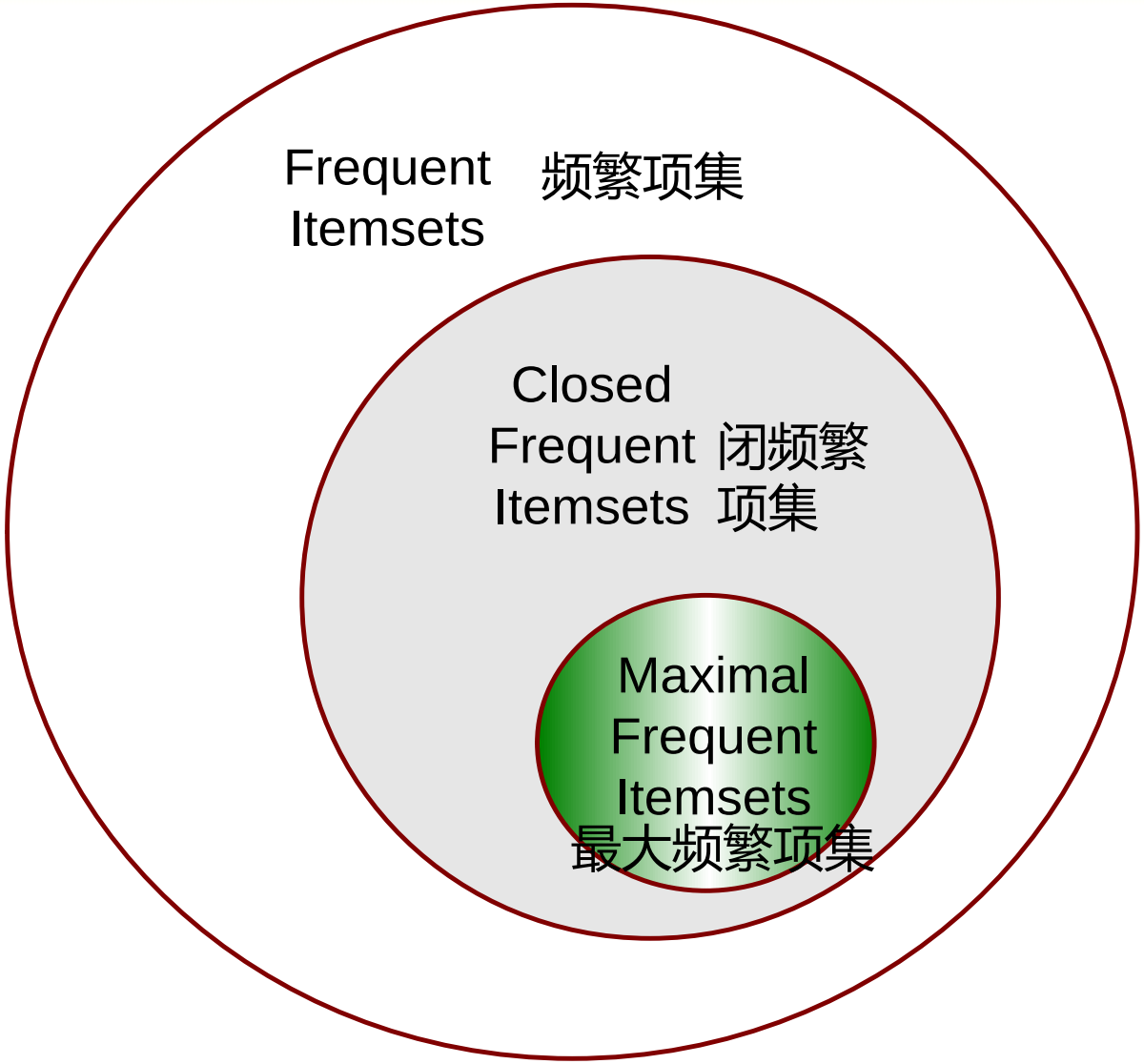
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	2
{A,B,C,D}	2

闭频繁项集和极大频繁项集



TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

闭频繁项集和极大频繁项集



闭频繁项集和极大频繁项集

- 假设事务数据库中只有两个事务：
  - DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
- 设最小支持度阈值 min_sup = 1：
- 有两个频繁项集：
  - {a_1, …, a_{100}}，支持度为1
  - {a_1, …, a_{50}}，支持度为2
- 只有一个极大频繁项集：
  - {a_1, …, a_{100}}，支持度为1
- 我们不能断言{a1, …, a50}是极大频繁项集
  - 因为它有一个频繁的超集{a_1, …, a_{100}}
- 频繁模式的总数为：
  - {a_1}: 2, …, {a_1, a_2}: 2, …, {a_1, a_{51}}: 1, …, {a_1, a_2, …, a_{100}}: 1
  - 总共$2^{100} - 1$个

同济大学交通运输工程学院
COLLEGE OF TRANSPORTATION ENGINEERING
TONGJI UNIVERSITY

关联模式评估方法

Apriori算法

Apriori算法的步骤

- Apriori算法命名源于算法使用了频繁项集性质的先验（Prior）知识。
- Apriori算法将发现关联规则的过程分为两个步骤：
  - 通过迭代，检索出事务数据库中的所有频繁项集，即支持度不低于用户设定的阈值的项集；
  - 利用频繁项集构造出满足用户最小信任度的规则。
- 挖掘或识别出所有频繁项集是该算法的核心，占整个计算量的大部分。

频繁项集

- 为了避免计算所有项集的支持度（实际上频繁项集只占很少一部分），Apriori算法引入潜在频繁项集的概念。
- 若潜在频繁k项集的集合记为C_k，频繁k项集的集合记为L_k，m个项目构成的k项集的集合为C_k^m，则三者之间满足关系$L_k \subseteq C_k \subseteq C_k^m$。
- 构成潜在频繁项集所遵循的原则是“频繁项集的子集必为频繁项集”。

关联规则的性质

- 性质1：**频繁项集的子集必为频繁项集。**
- 性质2：**非频繁项集的超集一定是非频繁的。**
- Apriori算法运用性质1，通过已知的频繁项集构成长度更大的项集，并将其称为潜在频繁项集。
  - 潜在频繁k项集的集合C_k是指由有可能成为频繁k项集的项集组成的集合。
- 以后只需计算潜在频繁项集的支持度，而不必计算所有不同项集的支持度，因此在一定程度上减少了计算量。

Apriori: 一种候选产生-测试方法

- 频繁项集的任何子集必须是频繁的
  - 如果 {啤酒, 尿布, 花生} 是频繁的, {啤酒,尿布}也是
  - 每个包含 {啤酒,尿布,花生}的事务 也包含 {啤酒,尿布}
- Apriori 剪枝原则:
  - 如果一个项集不是频繁的, 将不产生/测试它的超集
- 方法:
  - 由长度为k的频繁项集产生长度为 $(k+1)$ 的候选项集, 并且
  - 根据数据库DB测试这些候选
- 性能研究表明了它的有效性和可伸缩性

Apriori算法举例

数据库

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

最小支持度: $Sup_{min} = 2$

C_1

第1次扫描

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

第2次扫描

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{A, B, C}
{A, B, E}
{B, C, E}

第3次扫描

L_3

Itemset	sup
{A, B, C}	1
{A, B, E}	1
{B, C, E}	2

Apriori算法：伪代码

C_k：长度为k的候选项集
L_k：长度为k的频繁项集

(1) L_1={频繁1项集};
(2) for ($k = 2; L_{k-1} = \emptyset; k++$) do begin
(3) C_k = apriori_gen(L_{k-1}); //新的潜在频繁项集
(4) for all *transactions* $t \in D$ do begin
(5) C_t = subset(C_k, t); //找出t中包含的潜在的频繁项
(6) for all *candidates* $c \in C_t$ do
(7) c.count++;
(8) end;
(9) $L_k = \{c \in C_k | \text{c.count} \geq \text{minsup}\}$
(10) end;
(11) Answer=$\cup_k L_k$

Apriori的重要细节

- **如何产生候选?**
  - 步骤1：Lk的自连接
  - 步骤2：剪枝
- **产生候选，例如**
  - L3={abc, abd, acd, ace, bcd}
  - 自连接：L3*L3
    - 由abc和abd生成abcd
    - 由acd和ace生成acde
  - 剪枝:
    - 删除acde：因为ade不在L3里
  - C4 = {abcd}

如何产生候选

假定 L_{k-1} 中的项集已排序（按字典序排序）

步骤 1：L_{k-1}自连接

procedure apriori_gen(L_{k-1}: frequent$(k-1)$itemset)

for each项集$l_1 \in L_{k-1}$

for each项集$l_2 \in L_{k-1}$

if $(l_1[1] = l_2[1]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$

且

then

{

$c = l_1 \bowtie l_2$；//连接步，产生候选

if has_infrequent_subset(c, L_{k-1}) then

delete c;

else add c to C_k

}

return C_k

如何产生候选

步骤 2：剪枝

procedure has_infrequent_subset(c: candidate k itemset,
L_{k-1}: frequent $(k-1)$ itemset)

//使用先验知识

for each $(k-1)$ subset s of c

  if $s \notin L_{k-1}$ then

    return **TRUE**

return **FALSE**

示例（支持度计数=2）

AllElectronics 数据库

TID	List of item_ID's
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

C_1

项集	支持度计数
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

比较候选支持度计数与最小支持度计数 →

L_1

项集	支持度计数
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

由L_1产生候选C_2 →

C_2

项集
{I1,I2}
{I1,I3}
{I1,I4}
{I1,I5}
{I2,I3}
{I2,I4}
{I2,I5}
{I3,I4}
{I3,I5}
{I4,I5}

扫描D，对每个候选计数 →

C_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I4}	1
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2
{I3,I4}	0
{I3,I5}	1
{I4,I5}	0

比较候选支持度计数与最小支持度计数

L_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

示例

比较候选支持度计数
与最小支持度计数

L_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

由L_2产生
候选C_3

C_3

项集
{I1,I2,I3}
{I1,I2,I5}

扫描D，对每
个候选计数

C_3

项集	支持度计数
{I1,I2,I3}	2
{I1,I2,I5}	2

比较候选支持度计数
与最小支持度计数

L_3

项集	支持度计数
{I1,I2,I3}	2
{I1,I2,I5}	2

示例：说明

- 连接：
  - $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
  - $\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} =$
  - $\{\{I1, I2, I3\}, \{I1, I3, I5\}, \{I1, I2, I5\}, \{I2, I3, I4\}, \{I2, I4, I5\}, \{I2, I3, I5\}\}$

L_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

- 使用Apriori性质剪枝
  - 频繁项集的所有子集必须是频繁的
  - $\{I1, I2, I3\}$的2项子集是$\{I1, I2\}, \{I1, I3\}, \{I2, I3\}$。$\{I1, I2, I3\}$的所有2项子集都是$L_2$的元素。因此，保留$\{I1, I2, I3\}$在$C_3$中。
  - $\{I1, I2, I5\}$的2项子集是$\{I1, I2\}, \{I1, I5\}, \{I2, I5\}$。$\{I1, I2, I5\}$的所有2项子集都是$L_2$的元素。因此，保留$\{I1, I2, I5\}$在$C_3$中。
  - $\{I1, I3, I5\}$的2项子集是$\{I1, I3\}, \{I1, I5\}, \{I3, I5\}$。$\{I3, I5\}$不是$L_2$的元素，因而不是频繁的。因此，从$C_3$中删除$\{I1, I3, I5\}$

由频繁项集产生关联规则

- 根据公式产生关联规则
- $confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}$
- 对于每个频繁项集l，产生所有非空子集
- 对于每个l的非空子集s，如果$\frac{support_count(l)}{support_count(s)} \geq min_conf$
- 则输出规则$s \rightarrow (l - s)$

由频繁项集产生关联规则

- $X = \{I1, I2, I5\}$，最小置信度阈值=70%
- X的非空子集包括$\{I1, I2\}$，$\{I1, I5\}$，$\{I2, I5\}$，$\{I1\}$，$\{I2\}$，$\{I5\}$
  - $\{I1, I2\} \Rightarrow I5$，confidence=2/4=50%
  - $\{I1, I5\} \Rightarrow I2$，confidence=2/2=100%
  - $\{I2, I5\} \Rightarrow I1$，confidence=2/2=100%
  - $I1 \Rightarrow \{I2, I5\}$，confidence=2/6=33%
  - $I2 \Rightarrow \{I1, I5\}$，confidence=2/7=29%
  - $I5 \Rightarrow \{I1, I2\}$，confidence=2/2=100%
- 输出关联规则：$\{I1, I5\} \Rightarrow I2$，$\{I2, I5\} \Rightarrow I1$，$I5 \Rightarrow \{I1, I2\}$

L_1

项集	支持度计数
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

频繁模式挖掘的挑战

- 挑战
  - 事务数据库的多遍扫描
  - 数量巨大的候选
  - 候选支持度计数繁重的工作量
- 改进Apriori的基本思想
  - 减少事务数据库的扫描遍数
  - 压缩候选数量
  - 便于候选计数

提高Apriori算法的方法

- Hash-based itemset counting（散列项集计数）
- Transaction reduction（事务压缩）
- Partitioning（划分）
- Sampling（采样）

基于散列(hash): DHP

- DHP算法生效于Apriori算法的剪枝步过程中。
  - 在第k次扫描时，生成每个事务的k+1项集，代入一个Hash函数中，生成一个Hash表，同时记录每个桶中元素个数。
- 当生成C_{k+1}时，对L_k*L_k自连接产生的结果先进行代入上述Hash函数若所落的该桶的计数小于最小支持阈值，则该元素必定不为频繁项集，故可以过滤掉，不放入C_{k+1}中
  - 所有具有相同Hash值的项的总个数小于最小支持阈值

基于散列(hash): 样例

- 假设最小支持度计数为2，即min_sup = 2
- 第一次扫描，生成1-项目候选集C1
  - C1 = {{A}， {B}， {C}， {D}， {E}}
- 统计支持度，得到对应L1
  - L1 = {{A}， {B}， {C}， {D}， {E}}
- 对每个事务生成所有2项集
- 构造Hash函数
  - hash(x,y) = (order(x)*10 + order(y)) % 7，如order(A) = 1 , order(B) = 2

TID	Items
T1	A D E
T2	B D
T3	B D E
T4	C E
T5	C D
T6	C E
T7	A C D E
T8	C D E

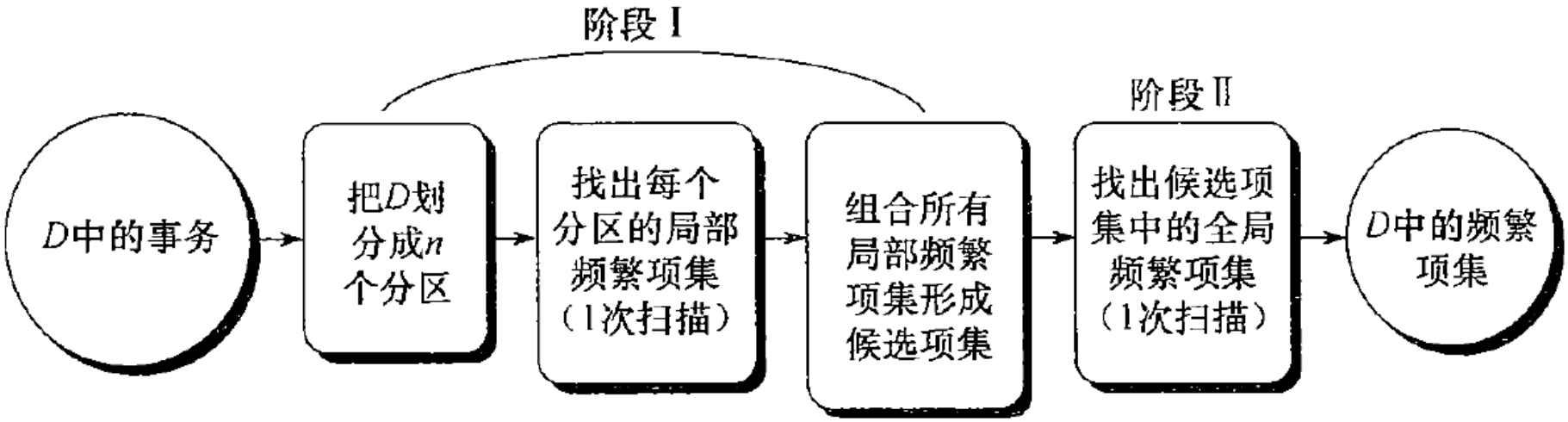
地址	0	1	2	3	4	5	6
元素	{A,D} {C,E} {C,E} {C,E} {A,D} {C,E}	{A,E} {A,E}	✖	{D,E} {B,D} {B,D} {D,E} {D,E} {D,E}	{B,E} ✖	✖	{A,C} {C,D} {C,D} {C,D}
计数	6	2	0	6	1	0	4

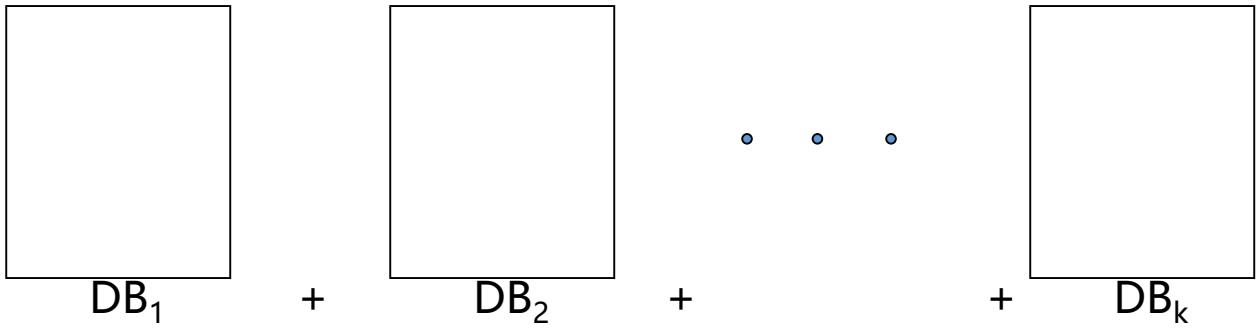
- L1 * L1 ={ {A, C}, {A, D}, {A, E}, {B, D}, {B, E}, {C, D}, {C, E}, {D, E} }
- 得到C2 = { {A, C}, {A, D}, {A, E}, {B, D}, {C, D}, {C, E}, {D, E} }

划分: 只扫描数据库两次

- 任何在数据库（DB）中频繁的项集必须必须至少在DB的一个划分中是频繁的
  - 扫描 1: 划分数据库, 并找出局部频繁模式（local frequent itemset）
  - 扫描 2: 求出全局频繁模式

划分: 只扫描数据库两次





$DB_1 + DB_2 + \cdots + DB_k = DB$

$sup_1(i) < \sigma DB_1$ $sup_2(i) < \sigma DB_2$ $sup_k(i) < \sigma DB_k$

抽样-频繁模式

- 选取原数据库的一个样本, 使用Apriori算法在样本中挖掘频繁模式
- 扫描一次数据库, 验证在样本中发现的频繁模式.
- 再次扫描数据库, 找出遗漏的频繁模式
- 牺牲一些精度换取有效性。

小结

- 频繁项集的定义及其挖掘的意义
- 关联规则的表达形式，支持度、置信度的计算方法
- 极大频繁项集、闭频繁项集的定义和判别方法
- APRIORI算法的原理和计算步骤
- 参考资料
  - 《数据挖掘概念与技术》第六章（重点：6.1-6.2.3）

同济大学交通运输工程学院
COLLEGE OF TRANSPORTATION ENGINEERING
TONGJI UNIVERSITY

第八讲 结束